Université Nationale du Vietnam, Hanoï, (UNV)
Institut Francophone pour l'Informatique, IFI

Rapport De TP

GÉNIE LOGICIEL AVANCÉ

Sujet : Réalisation d'un Petit Gestionnaire de Carnet d'Adresse

Réalisé par *:*
Ginel DORLEON
Promo 20, IFI, Juin 2016

Table des Matières

1 - INTRODUCTION

Pour l'introduction au cours **Génie Logiciel Avancé**, nous réalisé ce TP qui consiste à créer un G***estionnaire de Carnet d'Adresse***. A travers ce travail, nous appréhendons spécifiquement les concepts de la modélisation avec UML et programmation orientée-objet avec Java. De plus, il nous fait familiariser à l' environnement de développement intégré (IDE) « libre » (Open Source) ECLIPSE.
L'application développée permettra à un utilisateur de gérer ses contacts grâce à un carnet d'adresse. Cet outil offre plusieurs avantages à l'utilisateur. Il lui évitera une gestion totalement manuelle de ses informations et lui permet aussi de gagner du temps soit dans la recherche soit dans la sauvegarde de ses contacts.

1.1 SPÉCIFICATION DE L'APPLICATION

Le projet réalisé est un ***Petit Gestionnaire de Carnet d'Adresse*** dont les entrées pour une personne sont:

- Nom
- Numéros de téléphone
- Adresses (travail et domicile).

Ce ***Gestionnaire*** fournit à l'utilisateur les fonctionnalités suivantes:

- Créer une nouvelle personne/un nouveau contact;
- Chercher et afficher tous les numéros de téléphone et l'adresse d'une personne étant donné que son nom est entré à partir du clavier;
- Supprimer une personne de la liste étant donné que son nom est entré à partir du clavier;
- Chercher et modifier les numéros de téléphone et l'adresse d'une personne étant donné que son nom est entré à partir du clavier;
- Afficher la liste de toutes les personnes enregistrées dans la base de données (nom, numéros de téléphone, adresses)

(NB: Ici dans ce rapport, l'utilisation du terme personne est similaire de contact)

2 Exigences Fonctionnelles et Non-Fonctionnelles de l'Application

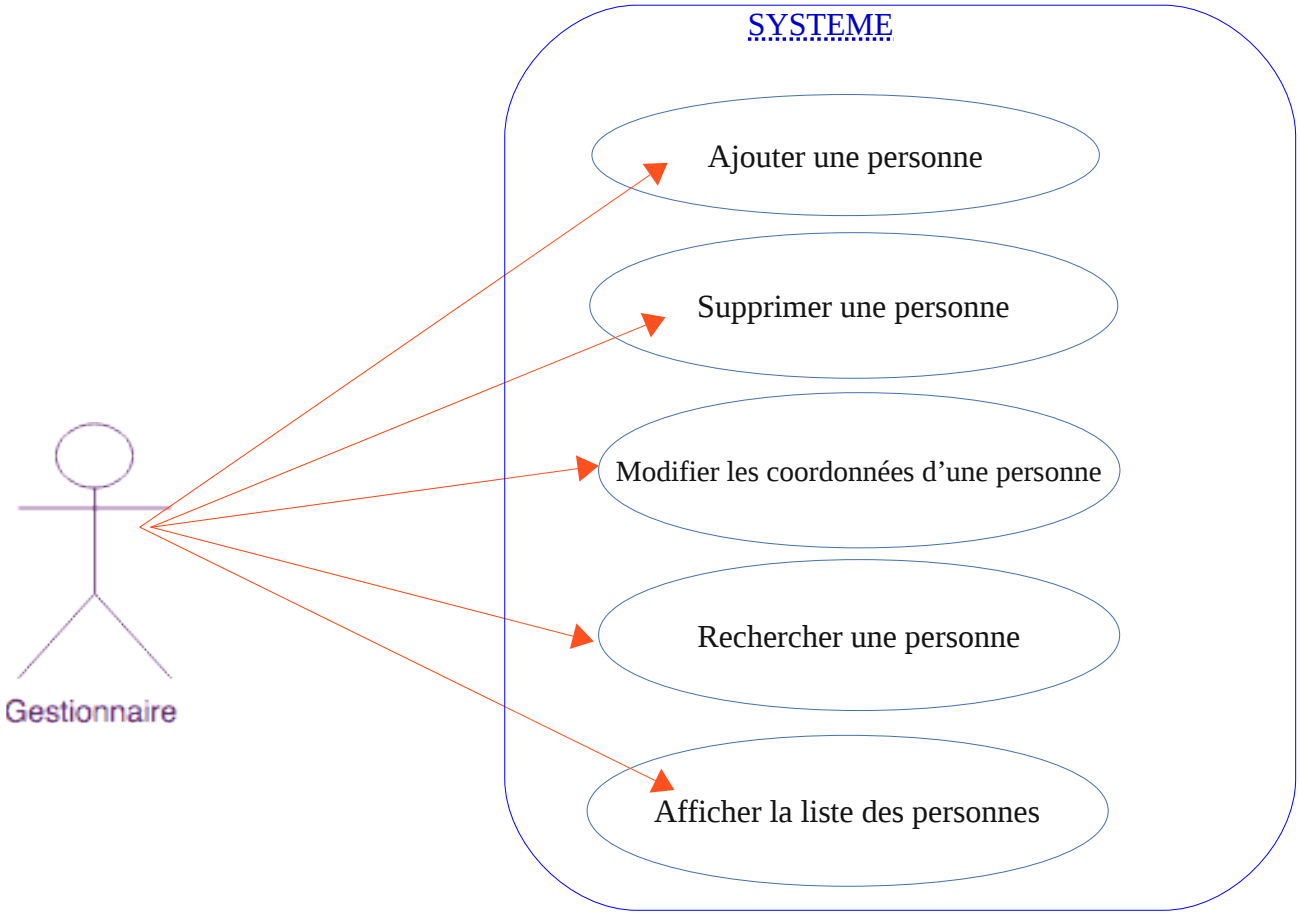
2. 1 Exigences Fonctionnelles (EF)

Par définition, les Exigences Fonctionnelles (EF) décrivent ce que le système doit pouvoir faire en terme d'actions et d'attentes. Dans le cadre de notre application, elles sont décrites comme suit :

EF I	**Ajout d'une personne**
	Lors de l'ajout d'une personne, le système doit pouvoir: -Demander de saisir les informations sur cette personne -Enregistrer les informations saisies
EF II	**Suppression d'une personne**
	Lors de la suppression d'une personne, le système doit pouvoir : -Demander une confirmation avant la suppression -Autoriser la suppression
EF III	**Recherche des coordonnées d'une personne**
	Le système doit pouvoir : -Permettre de rechercher les coordonnées d'une personne pourvu que son nom est entré via le clavier.
EF IV	**Modification des entrées**
	Le système doit pouvoir : - Permettre la modification d'une entrée même après la sauvegarde - Enregistrer les informations après la modification
EF V	**Affichage des entrées**
	Le système doit pouvoir : - Permettre l'affichage toutes les personnes avec leurs coordonnées respectives

a. Diagramme de Cas d'Utilisation

Les diagrammes de cas d'utilisation représentent généralement toutes les interactions des utilisateurs avec le système.



2.2 Les Exigences Non-Fonctionnelles (ENF)

Les Exigences Non-Fonctionnelles, ENF, caractérisent une propriété ou une qualité désirée du système telle que sa performance, sa robustesse, sa convivialité, sa maintenabilité, etc.

Dans le cadre de notre application, les ENF sont décrites comme suit :

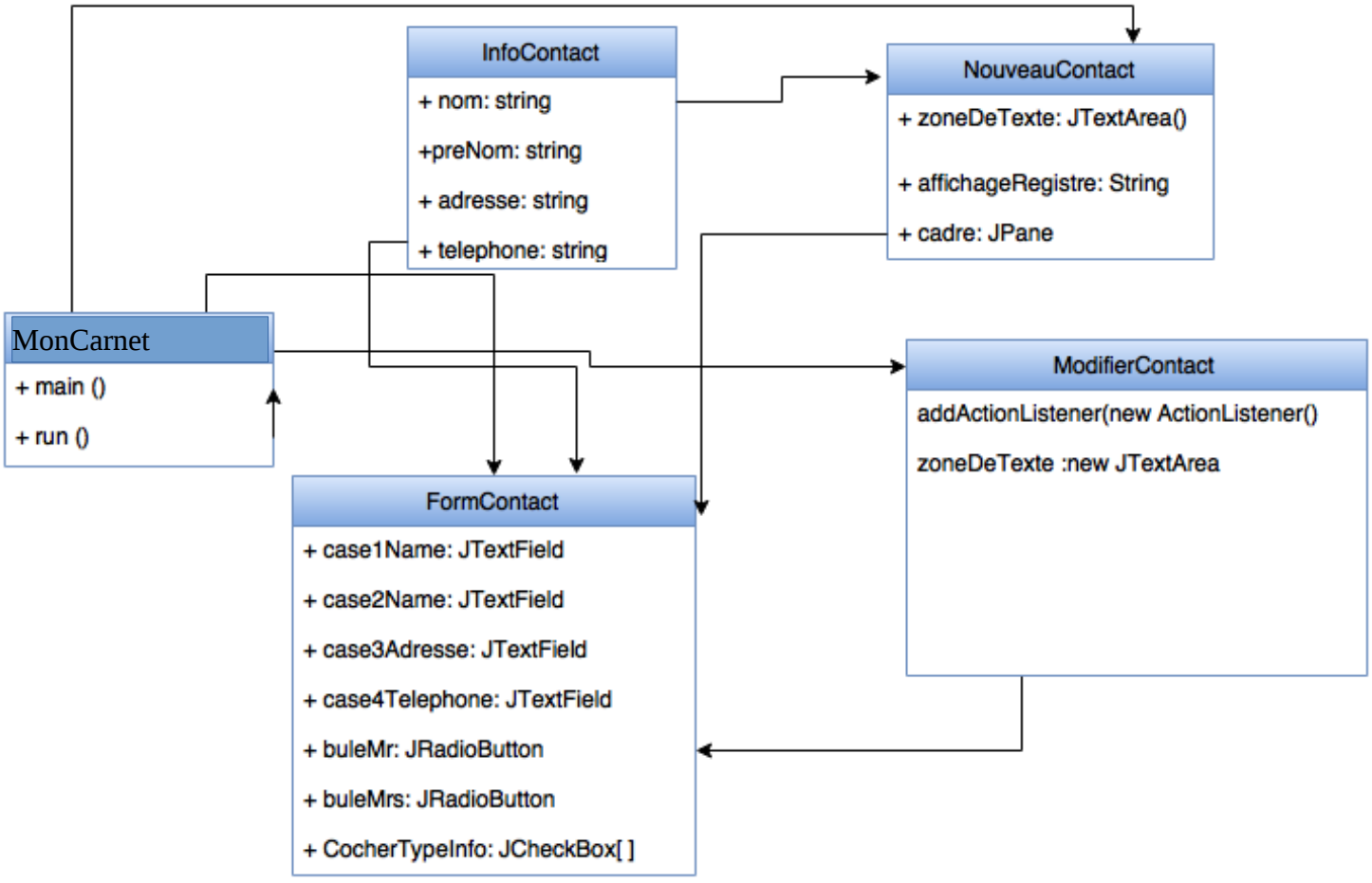
Exigences Non-Fonctionnelles	Description	Qualité / Attribut
ENF	Le temps de réponse du système doit être essentiellement court, maximum 2 secondes pour exécuter la requête de l'utilisateur	Performance
ENF	Notre application doit présenter des résultats précis et juste tels que attendus.. Elle doit être capable d' effectuer la bonne opération lorsque demandée.	Exactitude , Pertinence
ENF	En cas d'erreurs venant de l'utilisateur, les informations qui ont été déjà enregistrées ne doivent subir aucun changement.	Disponibilité
ENF	Les fonctions de l'application doivent être facilement comprises et interprétés par l'utilisateur.	Compréhension
	Le code source de l'application est bien documenté pour la facilité d'entretien et de mise à jour du système à l'avenir.	Faciliter la Maintenance

3. Conception de l'Application

3.1 Diagramme de Classe

Le diagramme de classe représente les classes constituant le système et les associations entre elles. Elle exprime de manière générale la structure statique du système, en termes de classe et de relations entre ces classes de notre application.

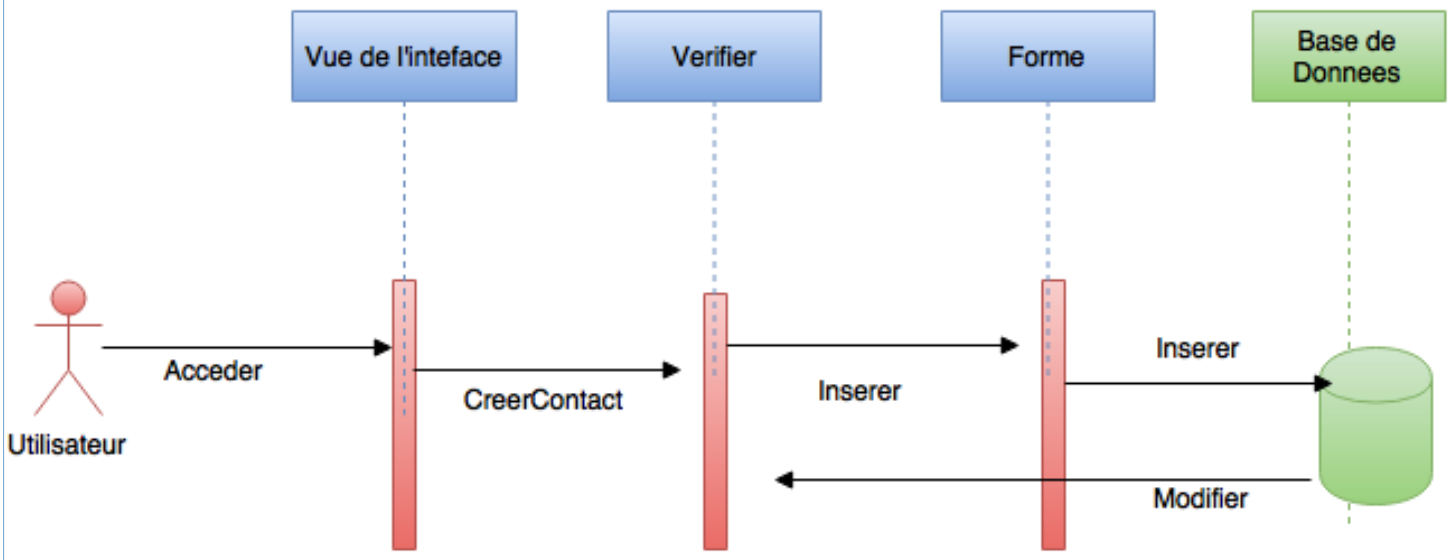
Voici une illustration de notre diagramme de classe ci-dessous



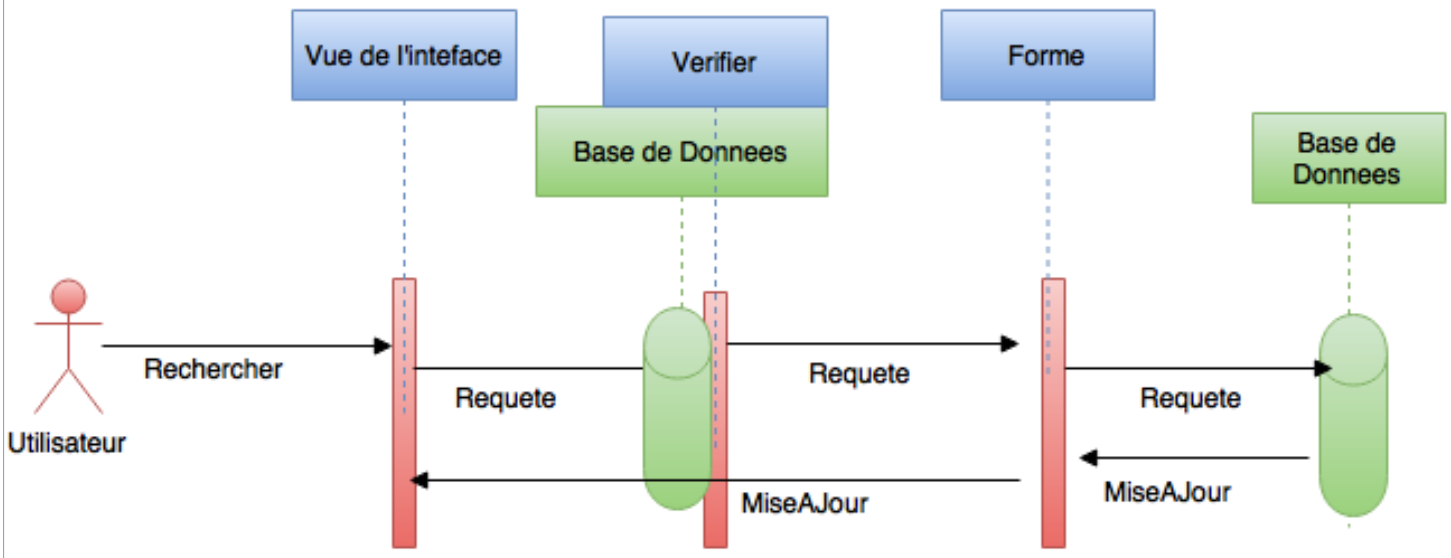
3.2 Diagramme de Séquence

Les diagrammes de séquences sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique

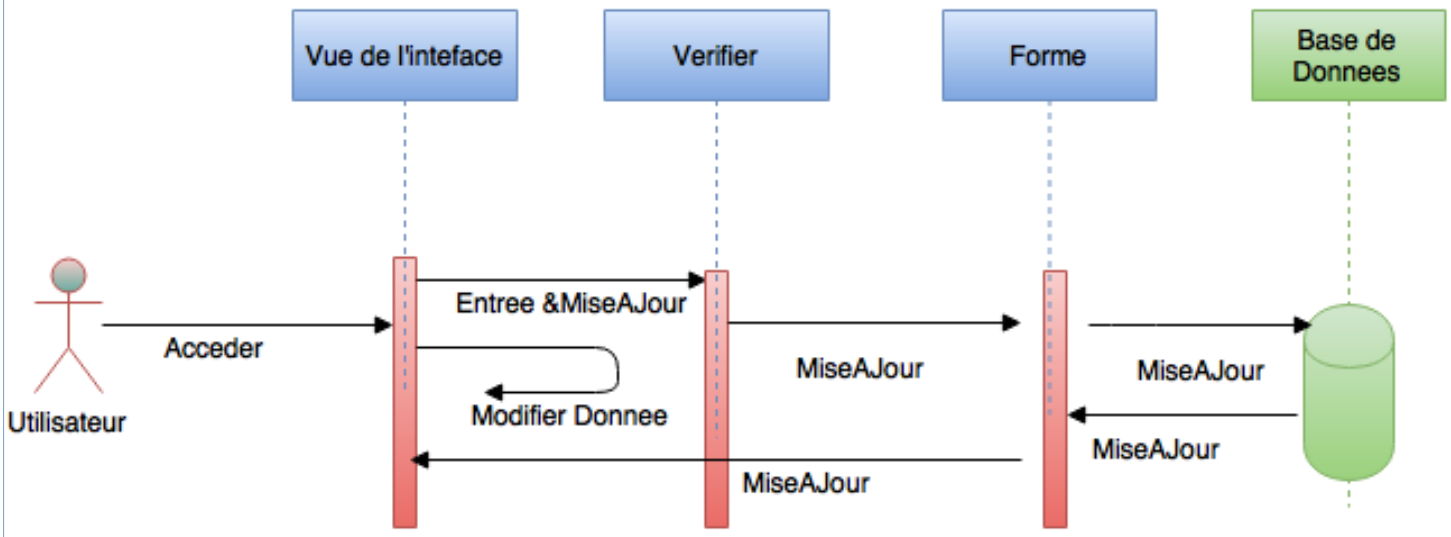
A- Créer un contact :(Diagramme de Séquence)



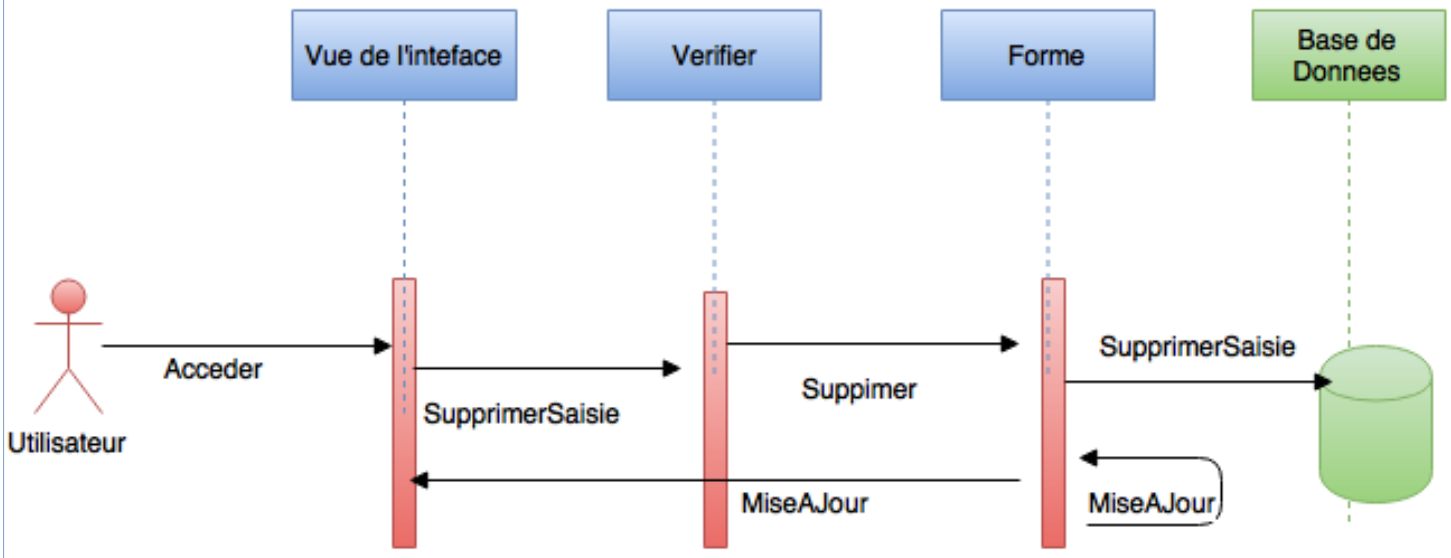
B-Rechercher un contact : Diagramme de Séquence



C-Modifier un contact : Diagramme de Séquence



D-4 Supprimer un contact : Diagramme de Séquence



4. IMPLÉMENTATION ET TEST

4.0 Environnement Matériel

Le développement de notre application a été réalisé sur une machine TOSHIBA Satellite

Un processeur AMD E1, 2.49 GHz.
Une mémoire vive de 2Go.
Un disque dur 250 Go.
Un écran 14 pouces.

4. 1 Environnement Logiciel

Le développement de l'application se fait exclusivement des plateformes libres.
L'implémentation se fait sous une machine Linux (Ubuntu), version 16.04, version Luna.
Utilisation de Eclipse comme IDE

4.2 Langage de programmation

L'une de nos contraintes ici c'est de programmer de façon orientée objet. Ainsi, le langage utilisé pour l'implémentation est le langage java ce qui de bien structurée de notre application.

4.3 La sauvegarde des données

Nous avons enregistrées nos données ici dans un fichier créer par le système. Ce fichier une fois créé peut être enregistré depuis l'interface utilisateur soit, sur le poste de travail, sur un support externe.

4.4 Méthode de test

Nous avons utilisé JUnit, pour faire le test unitaire et s'assurer d bon fonctionnement des méthodes utilisés, les variables et les classe utilisées

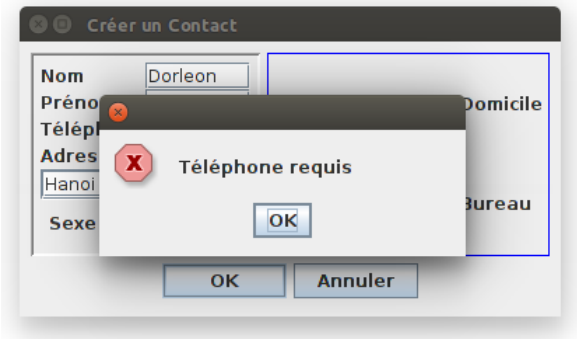
5. TEST D'ACCEPTATION

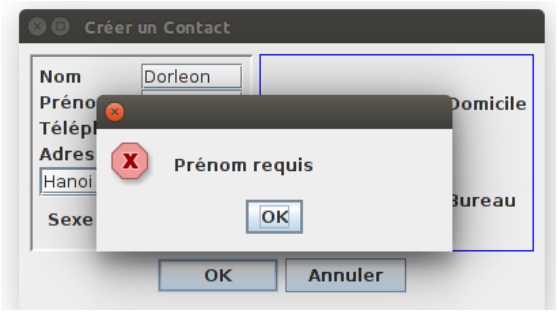
Voici ci-dessous quelques opérations suivant la spécification de l'application :

5.1 *Créer Nouveau Contact*

L' interface ci-dessous permet de saisir les coordonnées pour créer un nouveau contact.

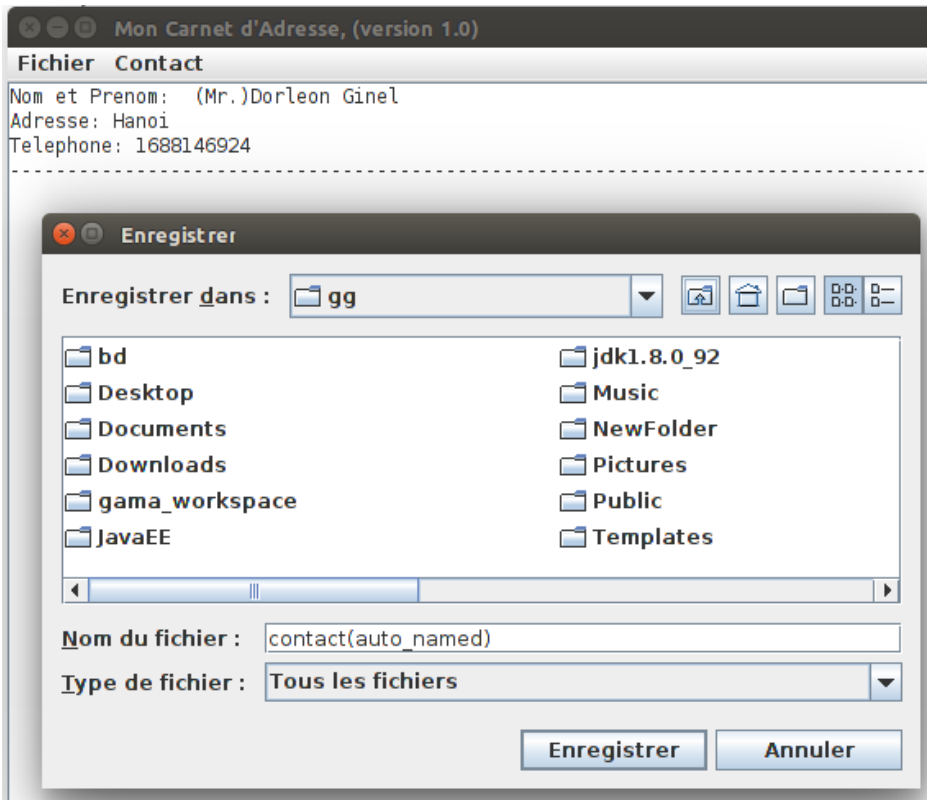
Lors de la création de nouveau contacts, tous les champs de saisie sont obligatoires; à défaut de ne pas saisir des informations pour un champ, un message avertissement apparaît empêchant la sortie et indiquant l'information manquante à saisir . (voir fig. ci-dessous)





5.2 Enregistrer un nouveau contact

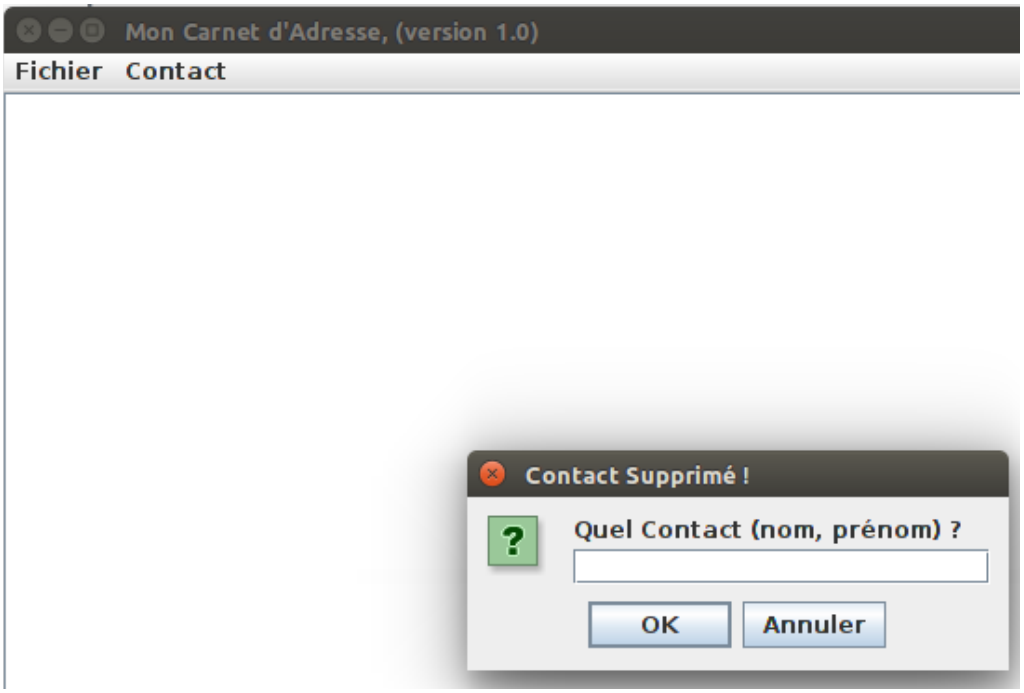
L'interface ci-dessous permet d'enregistrer et d'exporter ton fichier de contacts y compris toutes les données qui y sont rentrées.



Si on essaie de sortir sans avoir enregistré, un avertissement sera donné pour prévenir une perte involontaire des informations saisies (voir fig. ci-dessous).

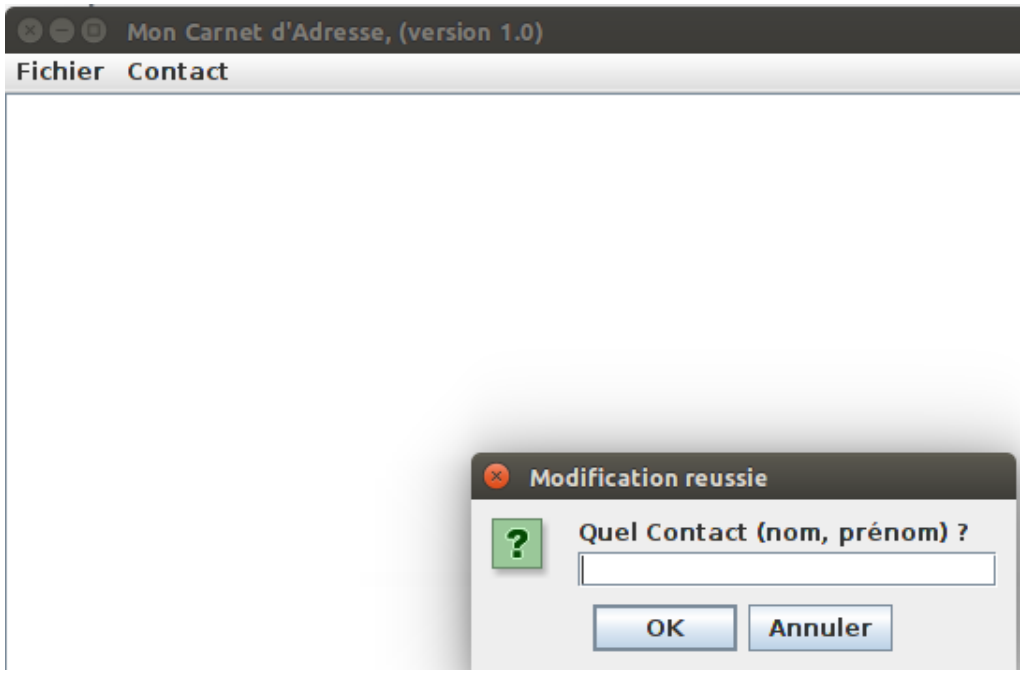
5.3 Supprimer un contact

Une fois lancée la commande « Supprimer Contact », une boite de dialogue invite à saisir les informations du contact à supprimer.
Pour supprimer , il faut saisir le nom et le prénom du contact en question séparé par une virgule. Une fois les informations saisies, le système lance la recherche pour trouver le contact et les informations relatives. Après avoir trouvé les informations, une autre boite de dialogue vous demande de confirmer la suppression.



5.4 Modification de contact

Une fois lancée la commande « Modifier Contact », une boite de dialogue invite à saisir les informations du contact à modifier.
Pour modifier , il faut saisir le nom et le prénom du contact en question séparé par une virgule. Une fois les informations saisies, le système lance la recherche pour trouver le contact et les informations relatives.



6. CONCLUSION

- Ainsi, c'est la description de l'application créée, un petit gestionnaire de carnet d'adresse d'où l'objet du TP .
La réalisation d'un tel TP nous a servi de rappel sur les concepts de la modélisation avec UML et programmation orientée- objet avec Java.
Ce travail n'est pas à sa plus haute perfection. Celle ci est à sa toute première version et nous comptons y améliorer pour une gestion d'autres coordonnées sur les contacts, comme groupe sanguin, date de naissance, pays , occupation etc...
Cependant, l'apport de ce travail a été d'une importance très considérable, en effet, il permet : de suivre une méthodologie de travail bien étudié, d'approfondir des connaissances sur les méthodes de développement des applications.

Informations Utiles à l'exécution du programme :

Pour lancer le programme, on doit donner le droit d'exécution au fichier MonCarnet.jar ;
Procéder comme suit :
- Clic droit sur MonCarnet.jar
- Propriétés
- Permissions
- Donner le droit de lecture et d'écriture à tous
- Cocher la case : Autoriser l'exécution du fichier comme programme.

Puis c 'est fini, le programme peut être lancé

Liens Utiles
Accéder aux cordes sources du projet sur github via ce lien :
- https://github.com/gdorleon/GenieLogiciel-TP1

CODES SOURCES

```
 1 package tp1GLogiciel;
 2
 3 import java.awt.Dimension;
30
31 /*
32  * Gestionaire d'Adresse.
33  *
34  *
35  */
36 public class LancementCarnet extends JFrame {
37
38     /**
39      *
40      */
41     public static final long serialVersionUID = 7394812270538140076L;
42
43     public static void main(String[] args) {
44         JFrame cadre = new LancementCarnet("Mon Carnet d'Adresse, (version 1.0)");
45         cadre.setLocationRelativeTo(null);
46         cadre.setVisible(true);
47         cadre.setSize(800, 600);
48     }
49
50     public LancementCarnet(String titre) {
51         super(titre);
52         setJMenuBar(barreDeMenus());
53
54         setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
55         addWindowListener(new WindowAdapter() {
56             public void windowClosing(WindowEvent arg0) {
57                 quitter();
58             }
59         });
60
61         zoneDeTexte = new JTextArea();
62         zoneDeTexte.setFont(new Font("Monospaced", Font.PLAIN, 12));
63         zoneDeTexte.setPreferredSize(new Dimension(400, 400));
64         setContentPane(new JScrollPane(zoneDeTexte));
65
```

```
66         pack();
67
68         registre = new TreeMap<>();
69     }
70
71     public JTextArea zoneDeTexte;
72     public Map<String, InfoContact> registre;
73     public boolean donneesSauvees = true;
74
75     public JMenuBar barreDeMenus() {
76         JMenuBar barreDeMenus = new JMenuBar();
77
78         JMenu menu = new JMenu("Fichier");
79         barreDeMenus.add(menu);
80
81         JMenuItem item = new JMenuItem("Afficher mes contacts:");
82         menu.add(item);
83         item.addActionListener(new ActionListener() {
84             public void actionPerformed(ActionEvent arg0) {
85                 menuFichierOuvrir();
86             }
87         });
88
89         item = new JMenuItem("Enregistrer Sous:");
90         menu.add(item);
91         item.addActionListener(new ActionListener() {
92             public void actionPerformed(ActionEvent arg0) {
93                 menuFichierEnregister();
94             }
95         });
96
97         menu.addSeparator();
98
99         item = new JMenuItem("Quitter");
100        menu.add(item);
101        item.addActionListener(new ActionListener() {
102            public void actionPerformed(ActionEvent arg0) {
103                quitter();
```

```
105         });
106
107         menu = new JMenu("Contact");
108         barreDeMenus.add(menu);
109
110         item = new JMenuItem("Creer un Contact");
111         menu.add(item);
112         item.addActionListener(new ActionListener() {
113             public void actionPerformed(ActionEvent arg0) {
114                 menuActionsCreationMembre();
115             }
116         });
117
118         item = new JMenuItem("Modifier un Contact");
119         menu.add(item);
120         item.addActionListener(new ActionListener() {
121             public void actionPerformed(ActionEvent arg0) {
122                 menuActionsModificationMembre();
123             }
124         });
125
126         item = new JMenuItem("Supprimer un Contact");
127         menu.add(item);
128         item.addActionListener(new ActionListener() {
129             public void actionPerformed(ActionEvent arg0) {
130                 menuActionsSuppressionMembre();
131             }
132         });
133
134         return barreDeMenus;
135     }
136
137     public void menuActionsCreationMembre() {
138         FormContact dial = new FormContact(this,
139                 "Créer un Contact");
140         dial.setLocationRelativeTo(this);
141         dial.setVisible(true);
142         if (!dial.getBonneFin())
```

```
144
145         String cle = cle(dial.getNom(), dial.getPrenom());
146         if (registre.containsKey(cle)) {
147             JOptionPane.showMessageDialog(this, cle + " est déjà dans vos contacts",
148                     "Erreur", JOptionPane.ERROR_MESSAGE);
149             return;
150         }
151
152         InfoContact membre = new InfoContact(dial.getNom(), dial.getPrenom(),
153                 dial.getAdresse(), dial.getTelephone(), dial.isSexeMasculin());
154         registre.put(cle, membre);
155         donneesSauvees = false;
156         affichageRegistre();
157     }
158
159     public void menuActionsModificationMembre() {
160         String cle = null;
161         int k;
162         for (;;) {
163             cle = JOptionPane.showInputDialog(this,
164                     "Quel Contact (nom, prénom) ?", "Modification de Contact",
165                     JOptionPane.QUESTION_MESSAGE);
166             if (cle == null)
167                 return;
168             k = cle.indexOf(',');
169             if (k >= 0)
170                 break;
171             JOptionPane.showMessageDialog(this,
172                     "Saisir le nom et le prénom séparés par une virgule", "",
173                     JOptionPane.ERROR_MESSAGE);
174         }
175         cle = cle(cle.substring(0, k), cle.substring(k + 1));
176
177         InfoContact membre = registre.get(cle);
178         if (membre == null) {
179             JOptionPane.showMessageDialog(null, cle
180                     + " : ce contact n'existe pas", "Erreur",
```